# Deep reinforcement learning using sparse distributed memory

Michal CHOVANEC

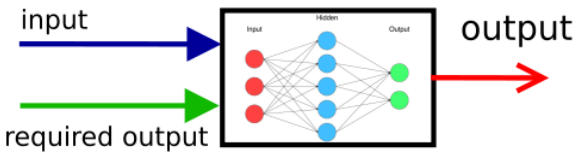
*October 2017*

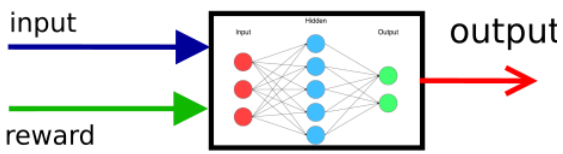
# Overview

- Reinforcement learning
- Sparse representation
- Advanced sparse distributed memory
- Experimental results

# Reinforcement learning

Supervised vs. Reinforcement learning

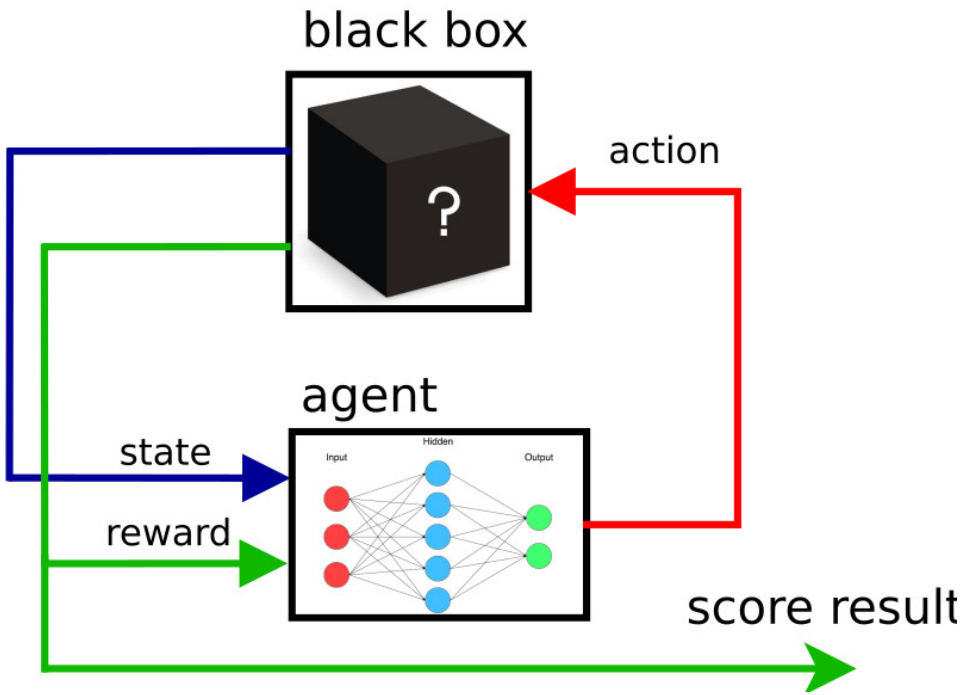



| | input | required output | output |
|---|---|---|---|
| supervised | yes | yes | class |
| reinforcement | yes | unknown | action |

# Reinforcement learning

Learning based on rewards and punishments



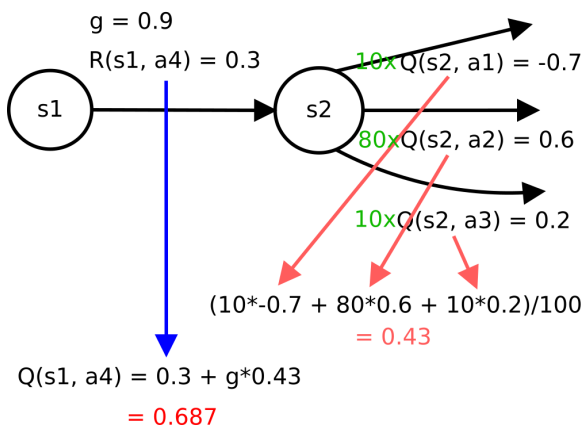

- input : state, reward
- output: action

# SARSA algorithm

State Action Reward State Action [1]

$$Q'(s,a) = (1-\alpha)Q(s,a) + \alpha(R(s,a) + \gamma Q(s',a'))$$

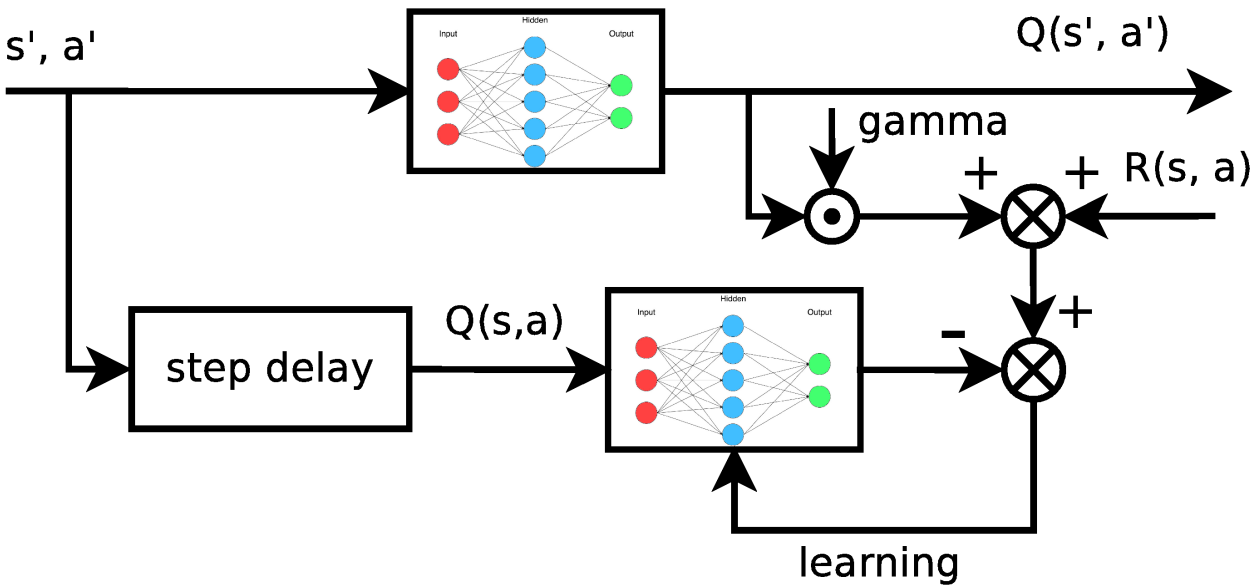
where
$Q(s,a)$ is previous state
$Q(s',a')$ is actual state
$R(s,a)$ is reward obtained in state $s$ after executing action $a$
$\gamma$ is discount factor $\gamma \in \langle 0,1 \rangle$
$\alpha$ is learning rate $\alpha \in (0,1)$




[1] Online Q-Learning using Connectionist Systems, by Rummery & Niranjan (1994)

# SARSA algorithm is recurrent

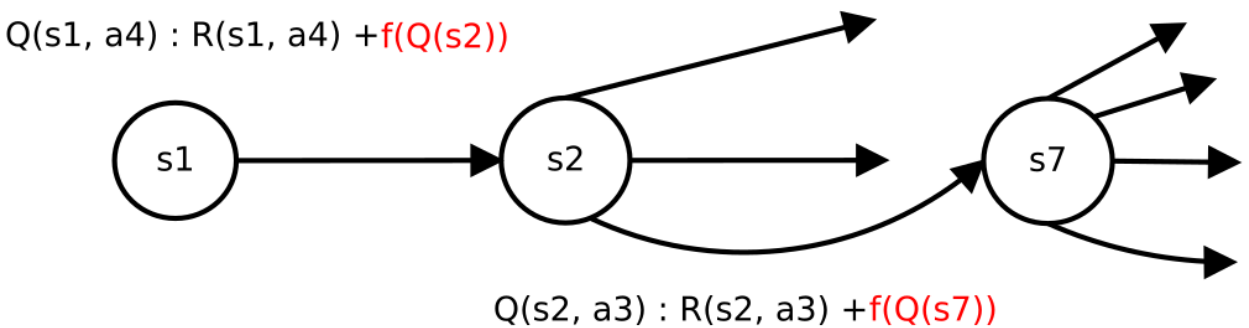
Main learning mechanism :
learn previous $Q(s, a)$ from actual $Q(s', a')$.



# SARSA algorithm is recurrent

Main learning mechanism :
learn previous $Q(s, a)$ from actual $Q(s', a')$.

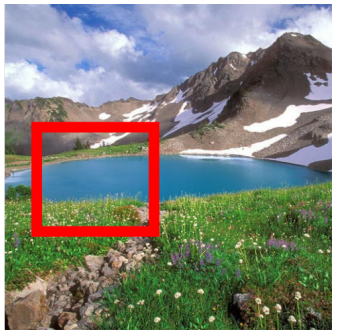
$$Q(n+1) = R(n+1) + \gamma Q(n+2)$$
$$Q(n) = R(n) + \gamma Q(n+1)$$
$$Q(n-1) = R(n-1) + \gamma Q(n) \text{// previous slide } [2]$$
$$Q(n-2) = R(n-2) + \gamma Q(n-1)$$



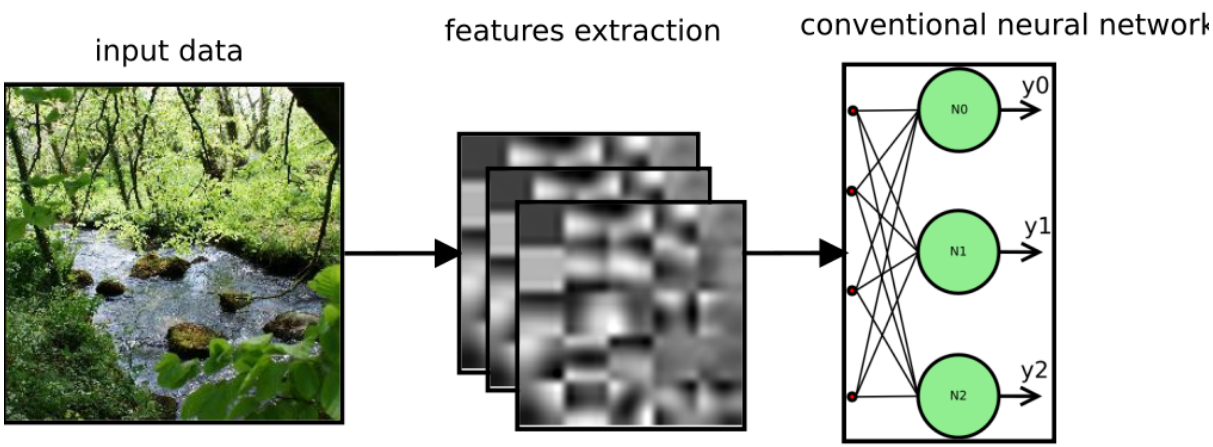

[1] equations are simplified, without $s$ and $a$

# Q value approximation

s =



- table
- linear combination of features
- neural network
- **sparse representation**



# Q value approximation

Linear combination of $K$ **learned** features $f_k(s)$, vector $v$ should be **sparse**, why ?

$$F(s) = \sum_{k=0}^{K-1} v_k f_k(s)$$

The reason for conjecturing that sparseness is an appropriate prior for the $a_i$ is based on the intuition that natural images may generally be described in terms of a small number of structural primitives—for example, edges, lines, or other elementary features (Field, 1994). In addition, one can see evidence for sparse structure in images by filtering them with a set of log-Gabor filters and collecting histograms of the resulting output distributions; these distributions typically show high kurtosis (Field, 1993), which is indicative of sparse structure.

---

[1] Olshausen and Field (1997): Sparse coding with an overcomplete basis set

[2] Olshausen BA, Field DJ (2004) : Sparse coding of sensory inputs. Current Opinion in Neurobiology, 14, 481-487

[3] Mushroom body, locust (Laurent)

[4] HVC, zebra finch (Fee)

[5] Auditory cortex, mouse (DeWeese & Zador)

[6] Hippocampus, ratprimate(Thompson & Best; Skaggs)

[7] Motor cortex, rabbit (Swadlow)

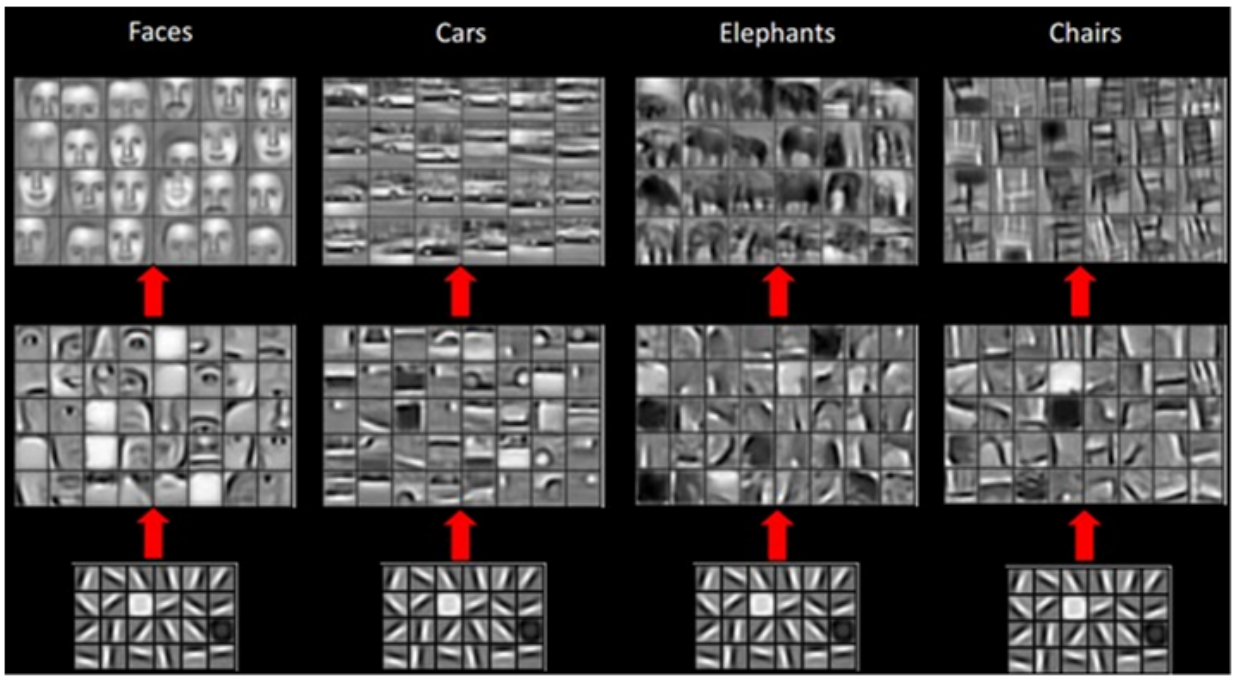
[8] Visual cortex, monkeycat (Vinje & Gallant)

[9] Inferotemporal cortex, human (Fried & Koch)

# Looking for symmetry

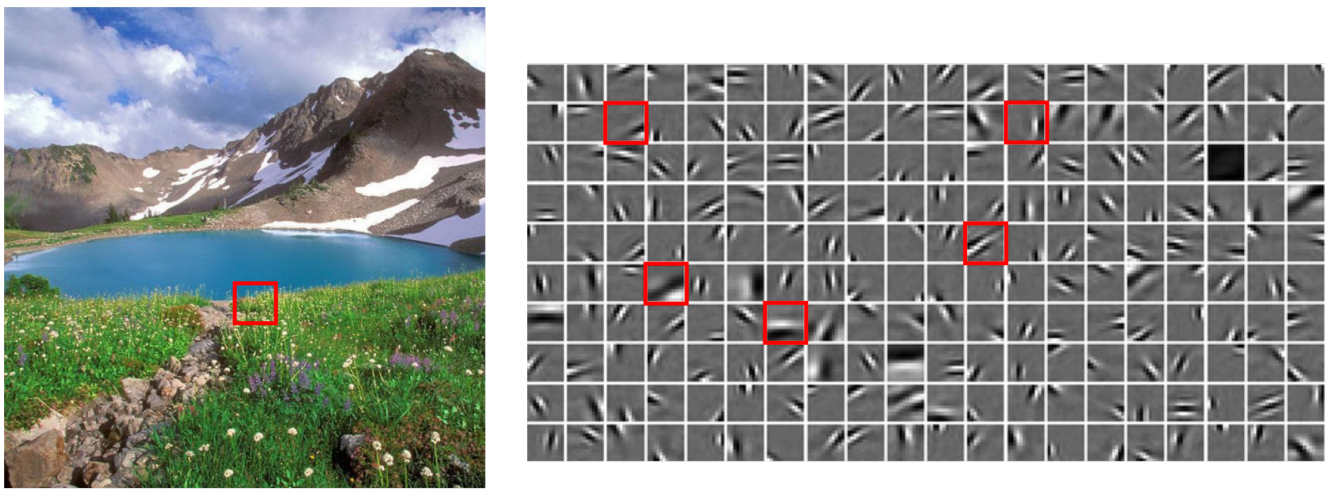
- Euler-Lagrange equation
- Noether's theorem
- Conservation law

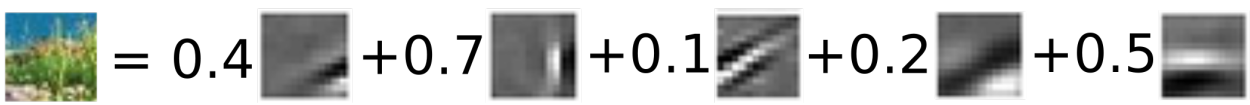
$$S(q) = \int_a^b L\left(t, q(t), \frac{dq(t)}{dt}\right) dt$$



# Sparse representation

$$F(s) = \sum_{k=0}^{K-1} v_k f_k(s)$$





# Sparse representation

$$E(w,y) = \sum_{k=0}^{K-1} \|x_k - wy_k\|_2^2 + \lambda_0 \|y_k\|_1^2 + \lambda_1 \|w\|_{2,1}^2$$

$$\text{subject to} \quad y_{ki} > 0$$

- reconstruction condition

$$\|x_k - wy_k\|_2^2$$

- $y_k$ sparsity condition

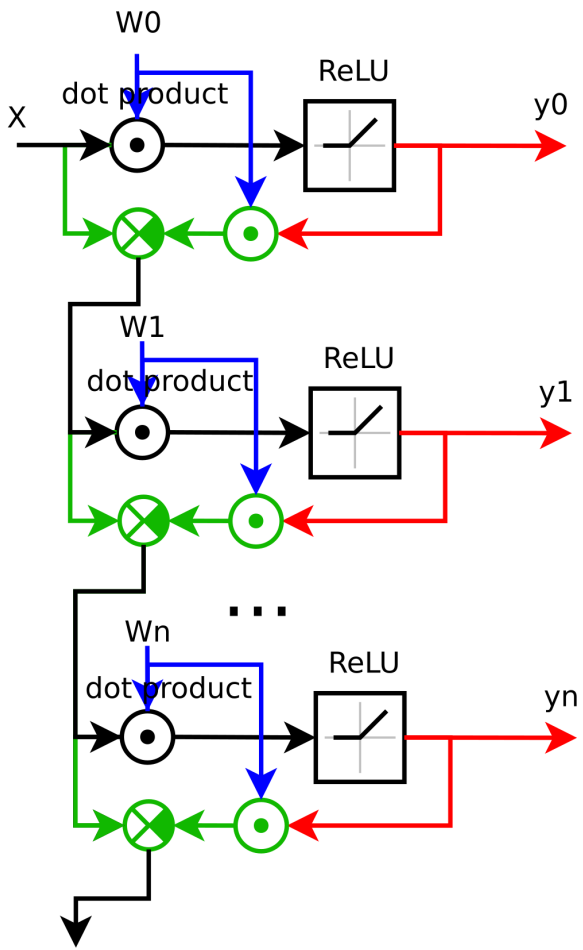
$$\lambda_0 \|y_k\|_1^2$$

- $w$ sparsity condition

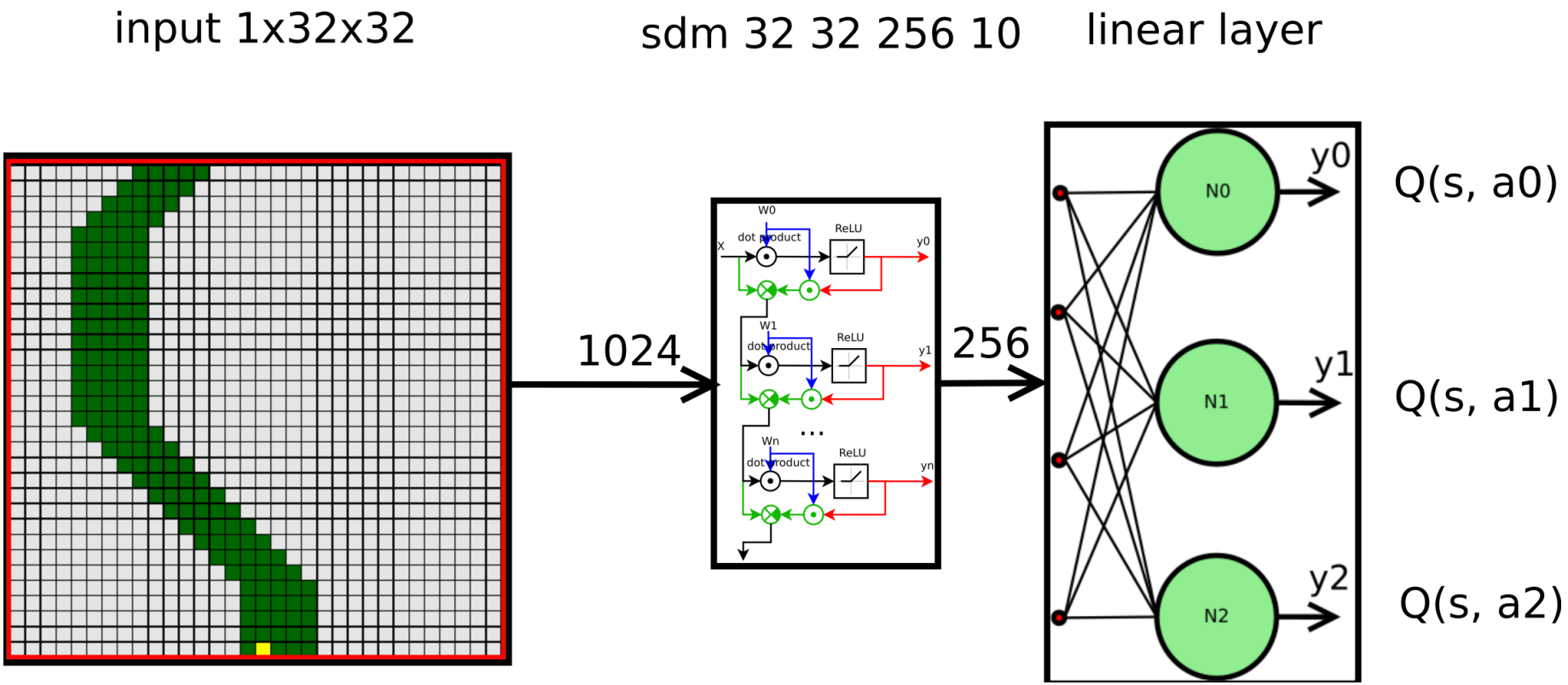
$$\lambda_1 \|w\|_{2,1}^2$$

# Estimating $W$ and $Y$ - modified K-SVD algorithm

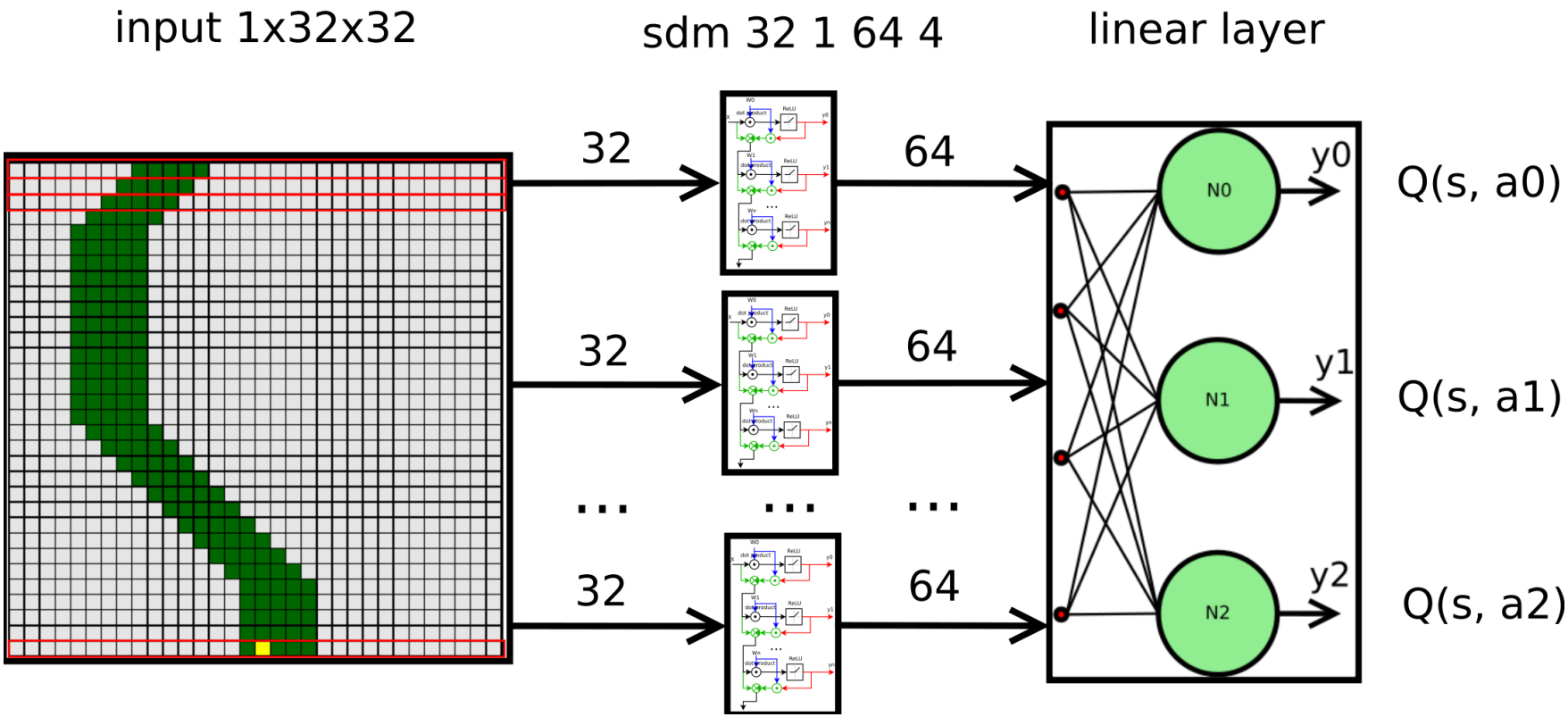
**Data:** W, depth, X
**Result:** sparse representation Y, features matrix weights W
initialization
$W \leftarrow random$, $R \leftarrow X$
$\eta$ learning rate, $\epsilon$ shrink value
**for** *j from 0 to depth* **do**
  $best = \arg\max_i W[i]R$
  $d = ReLU(\frac{W[best]R}{\|W[best]\|_2^2})$
  $Y[best] = d$
  **for** *i from 0 to size(R)* **do**
    $dw = d(R[i] - W[best][i])$
    $W[best][i] = W[best][i] + \eta dw$
    $W[best][i] = shrink(W[best][i], \epsilon)$
    R[i] = R[i] - d*W[best][i]
  **end**
**end**



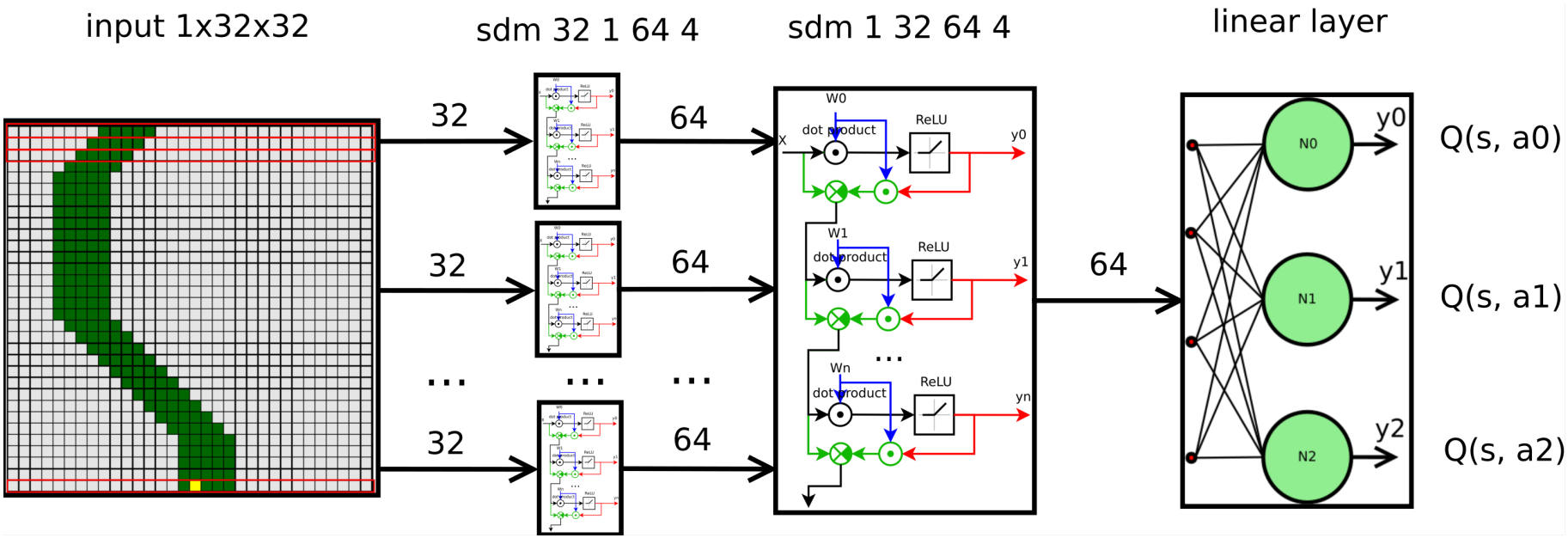
# Advanced sparse distributed memory



# Advanced sparse distributed memory + Convolution



# Advanced sparse distributed memory + Convolution + Deep layer

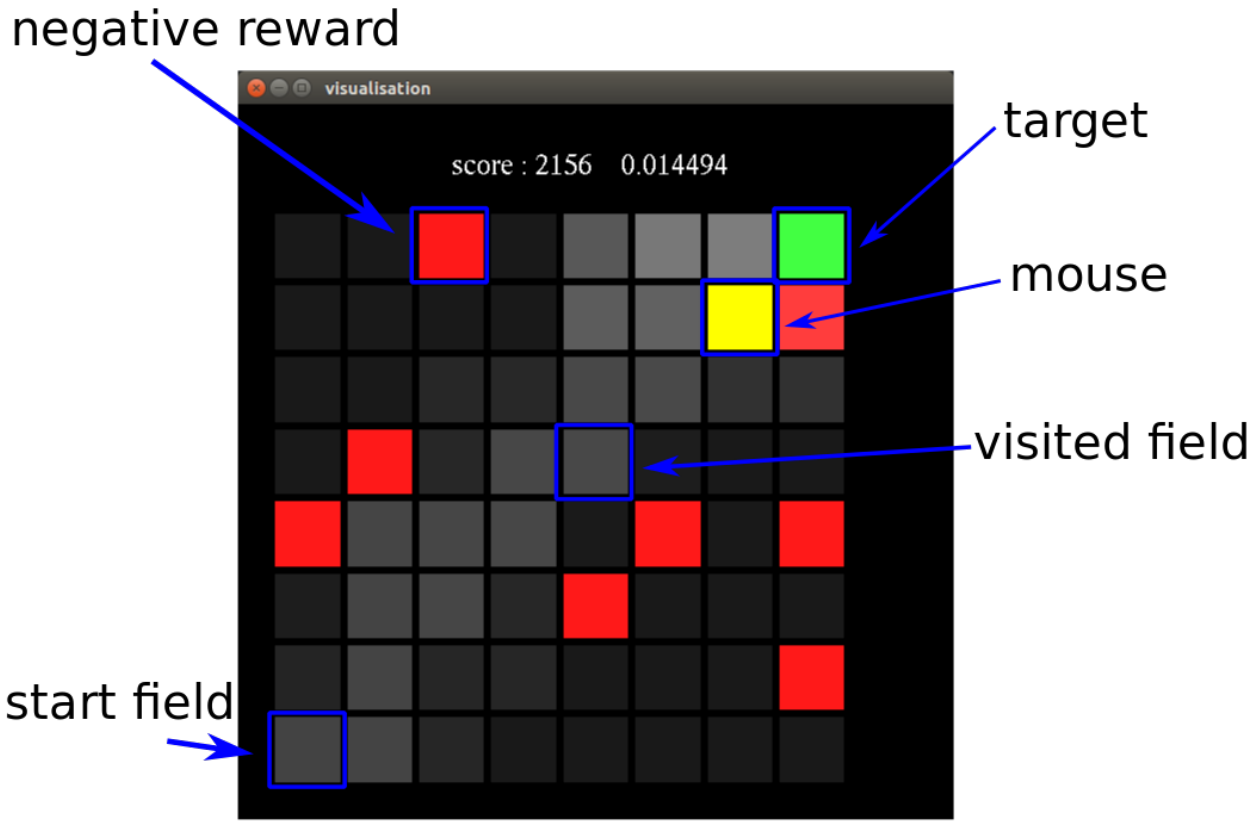


# Experimental results

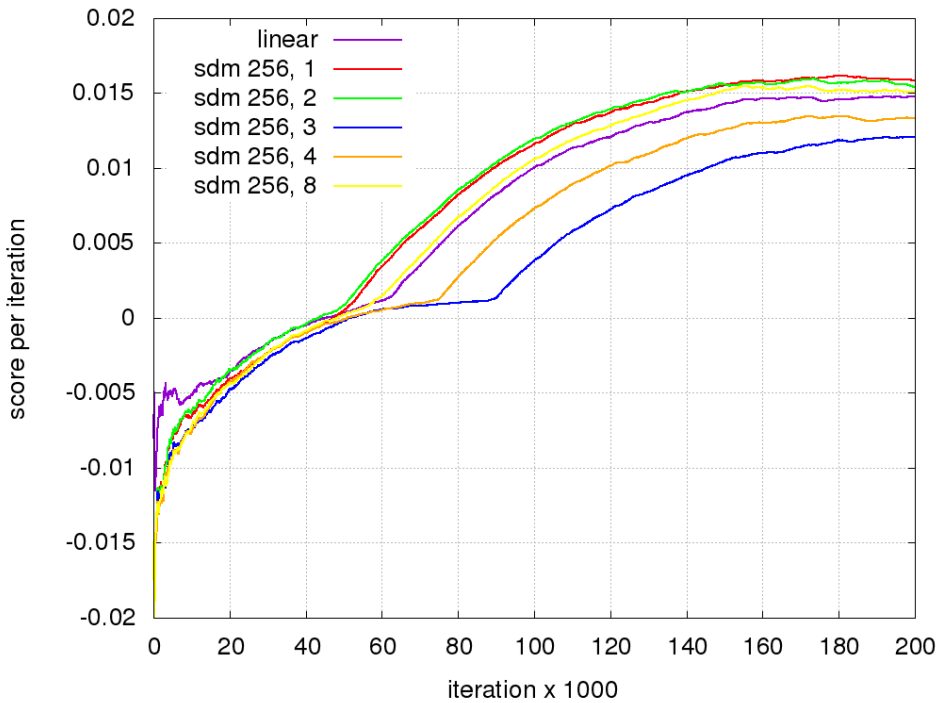
- Mouse and food
- Worms
- Line follower

# Mouse and food

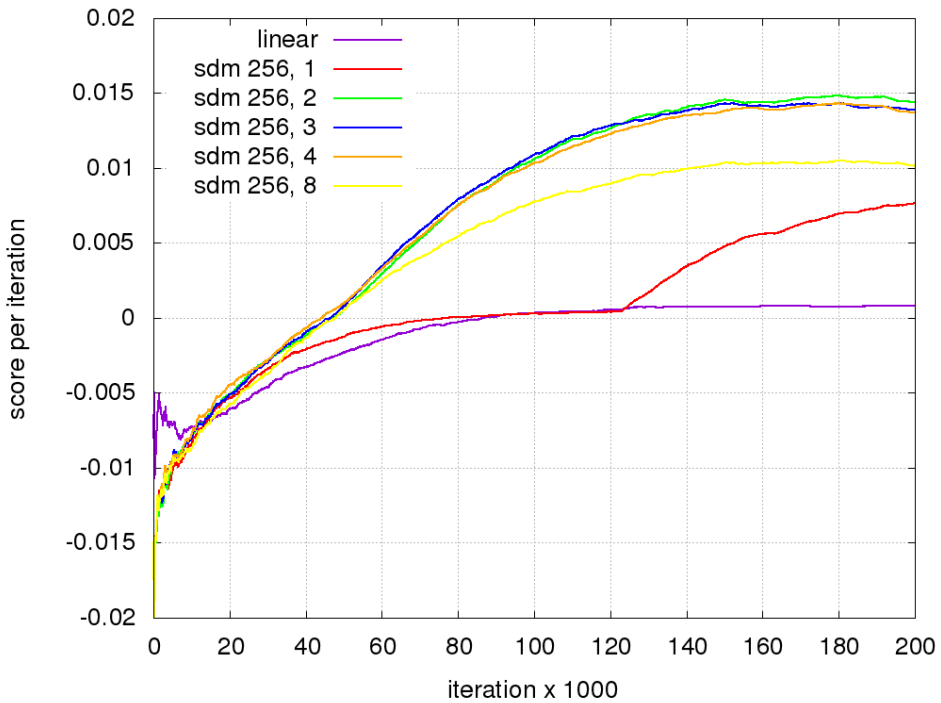
- state : vector $64x\{0,1\}$, 1 if mouse on field
- action : left, right, up, down
- reward : +1 for target hit, -1 for red hit



# Mouse and food

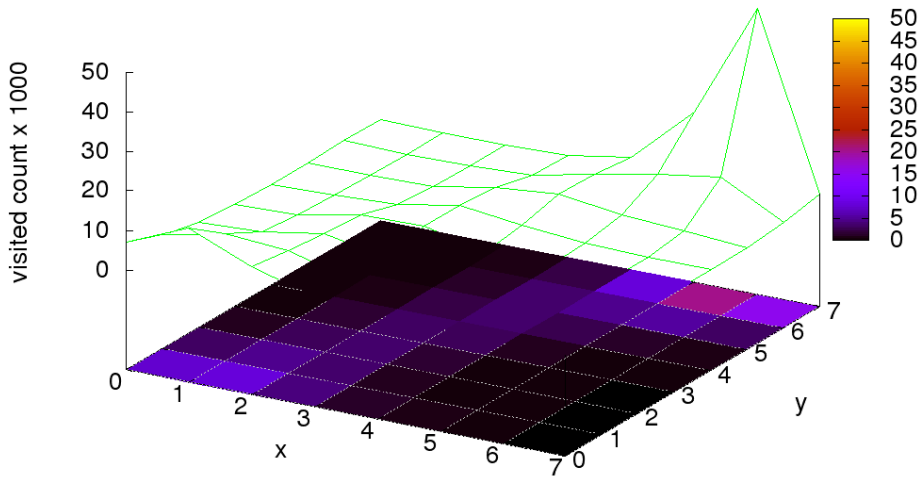



Progress rate, state noise 0.0

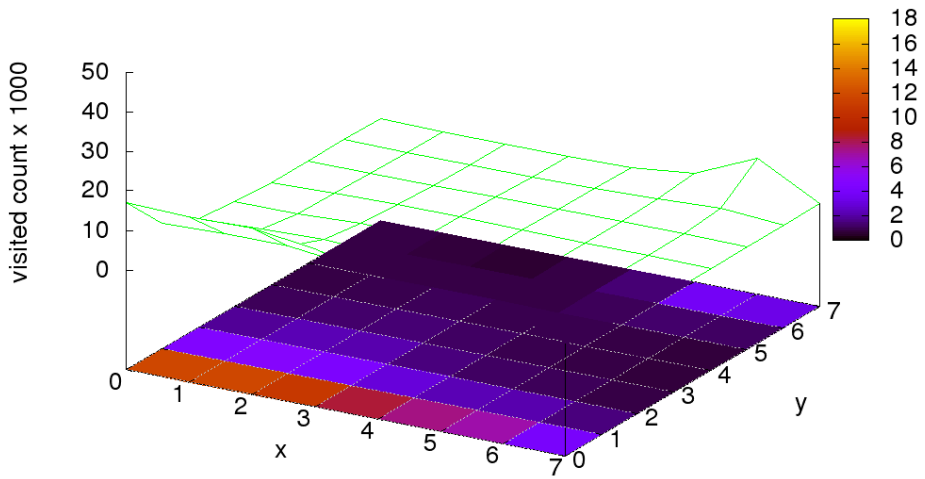



Progress rate, state noise 0.1

# Mouse and food - linear approximator

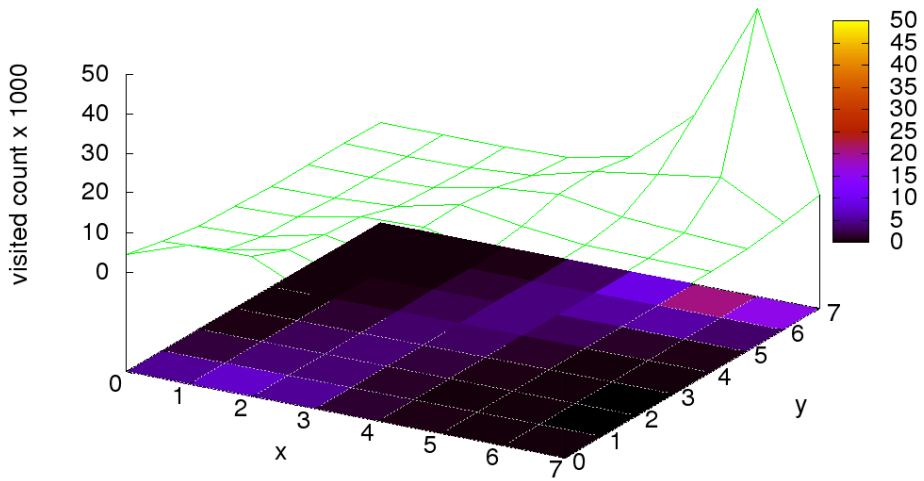



Visited field rate, state noise 0.0

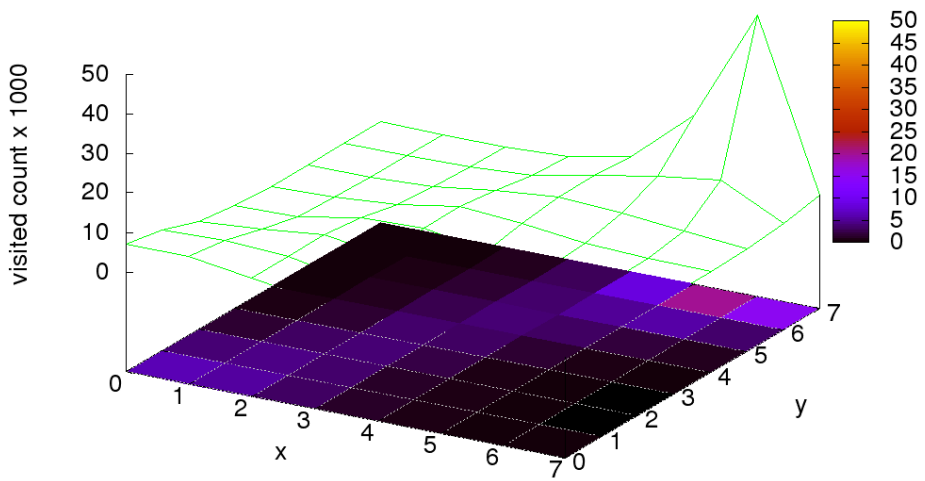



Visited field rate, state noise 0.1

# Mouse and food - sdm approximator 256, 2

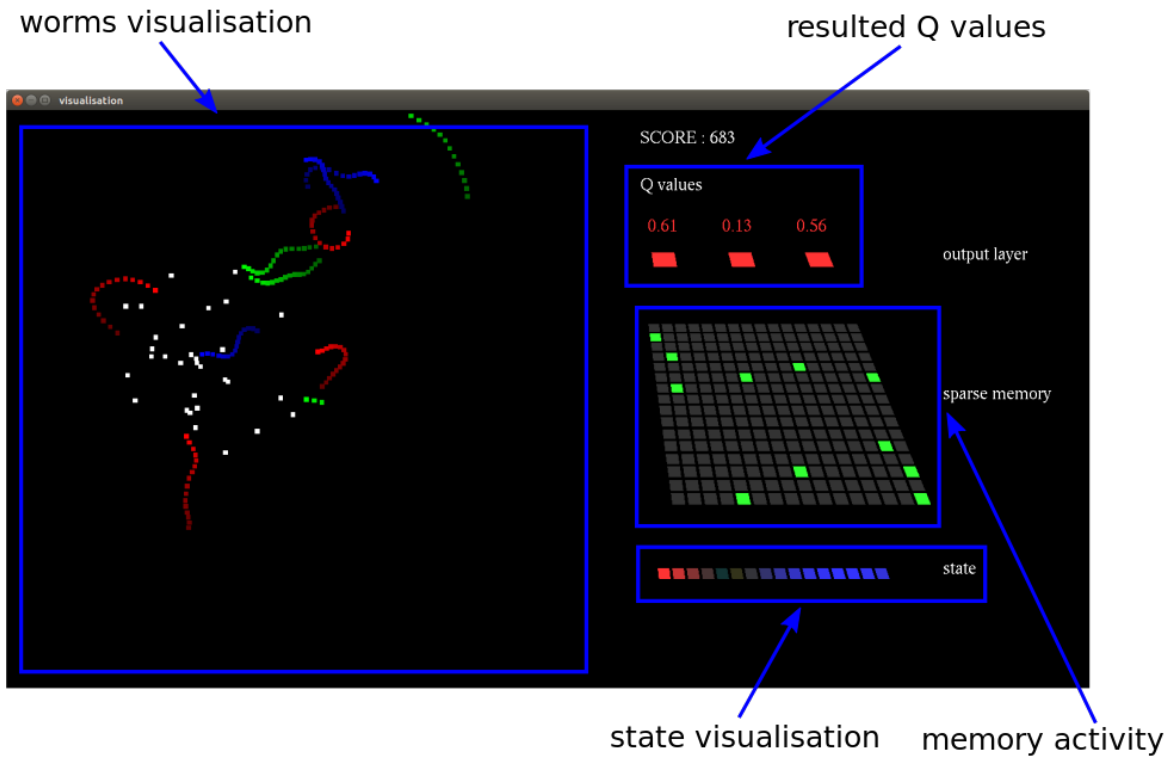



Visited field rate, state noise 0.0

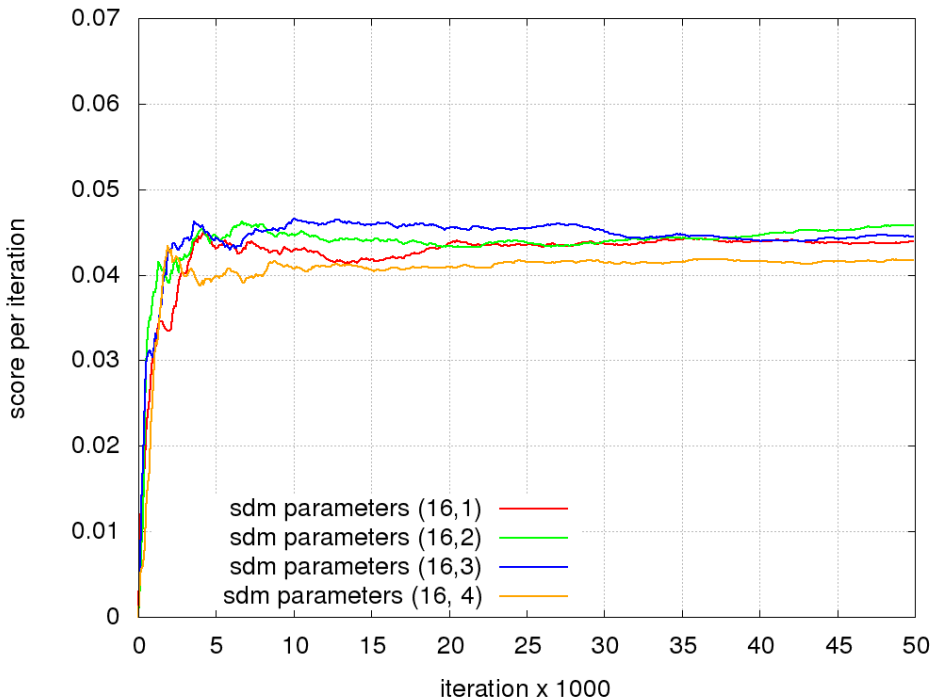


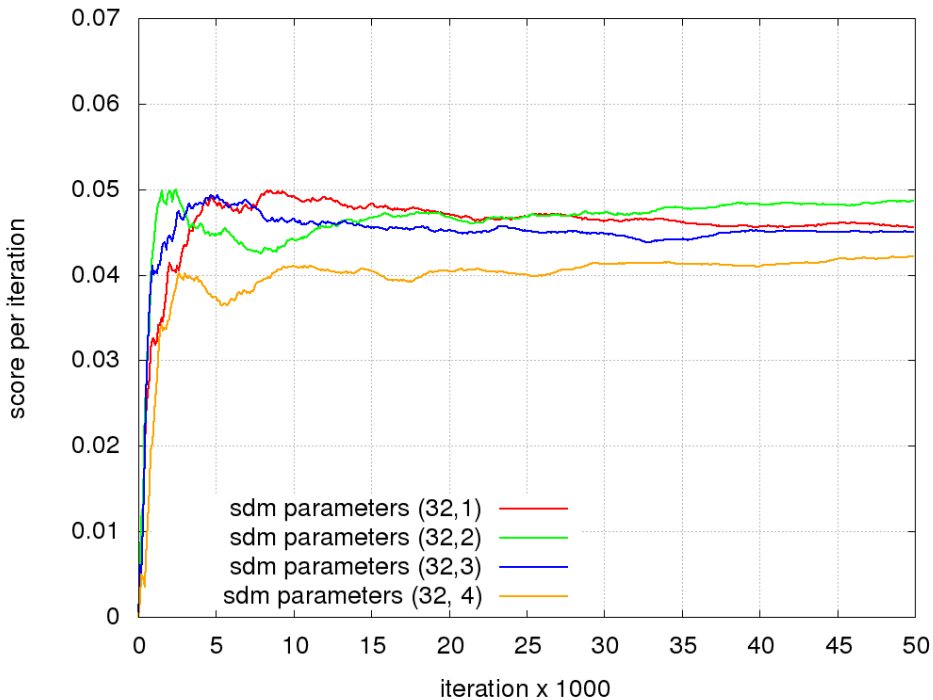

Visited field rate, state noise 0.1

# Worms

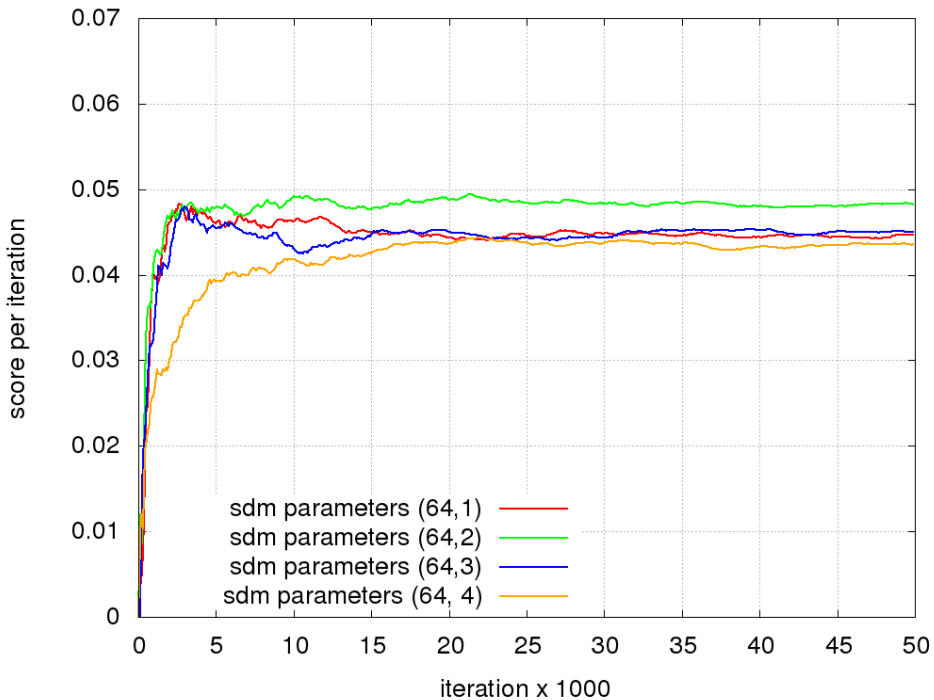

worms visualisation
resulted Q values
visualisation
SCORE : 683
Q values
0.61
0.13
0.56
output layer
sparse memory
state
state visualisation
memory activity

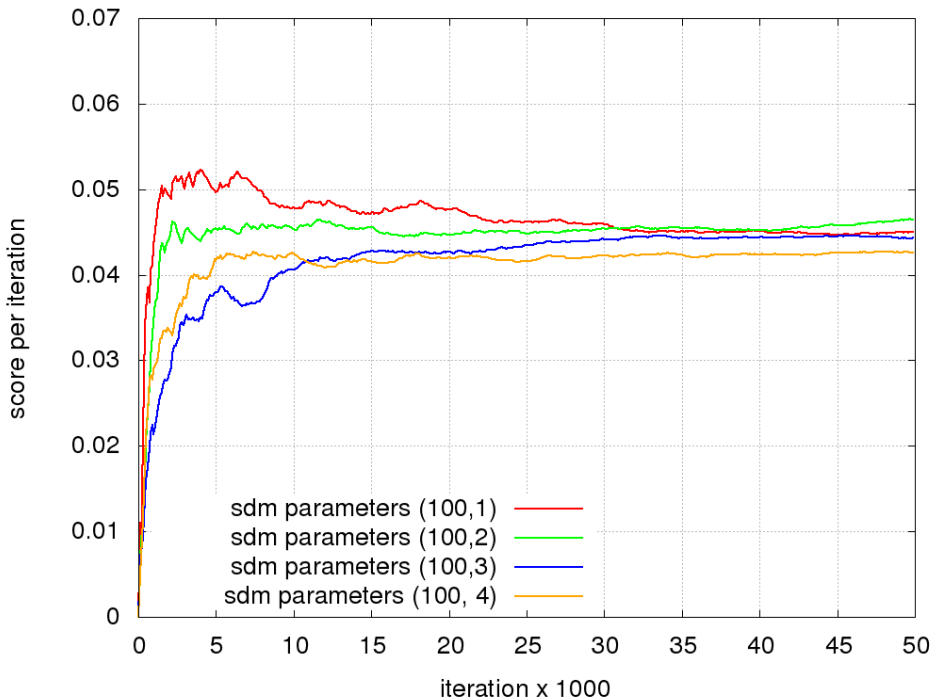
# Worms - memory size and activity results



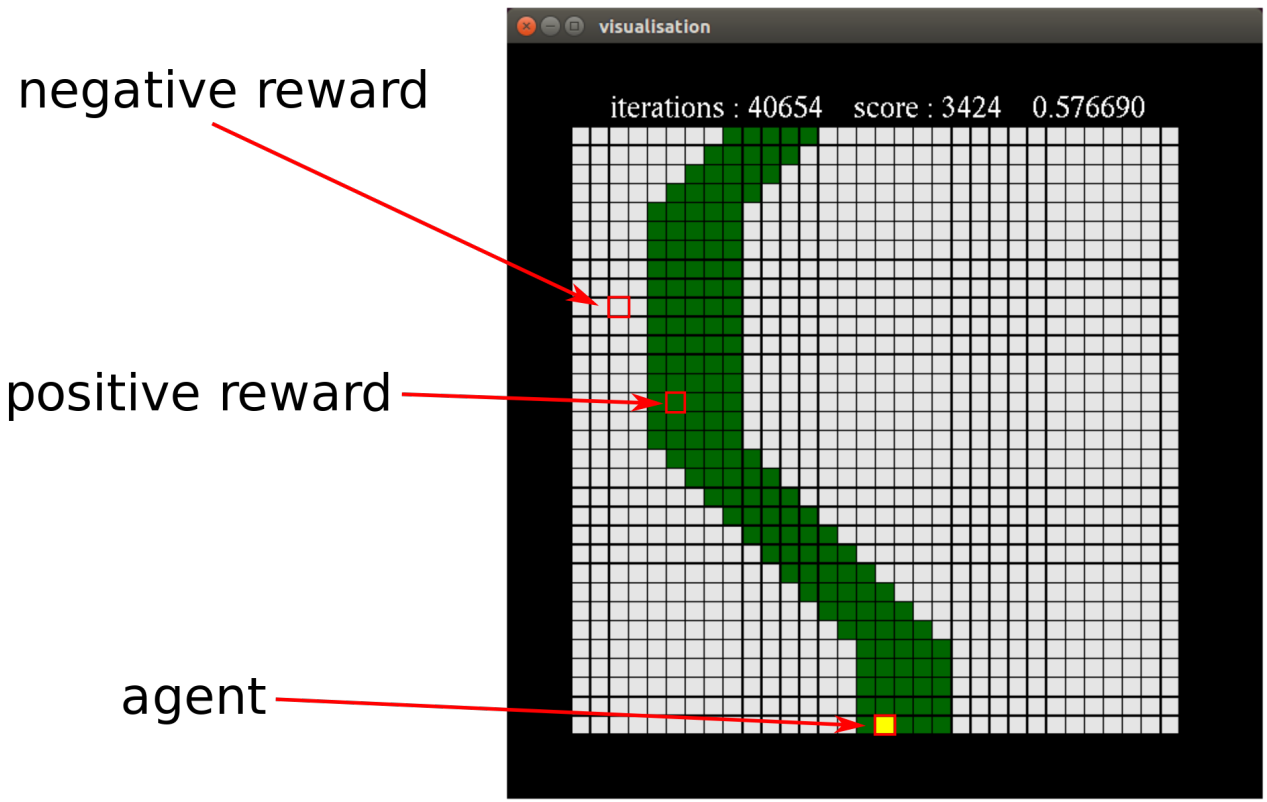



# Worms - memory size and activity results



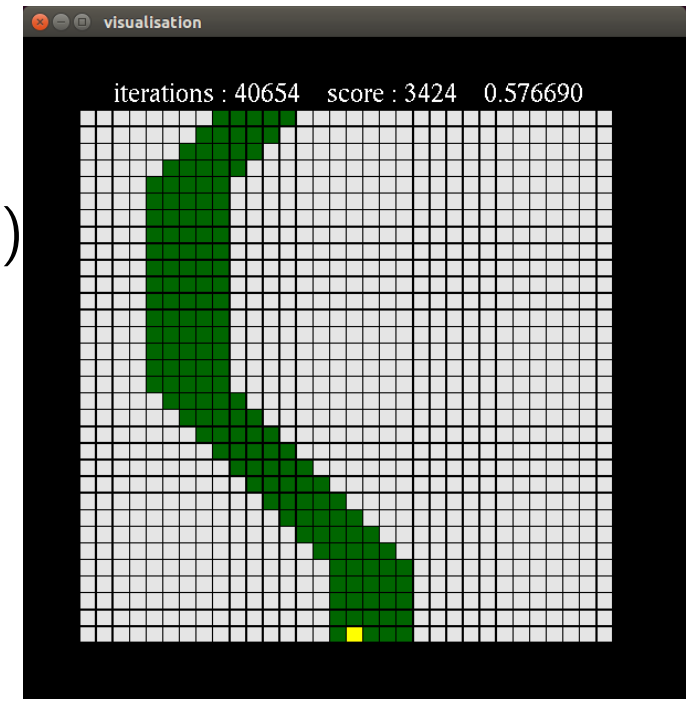
# Line follower



# Line follower

- motion equations of robot

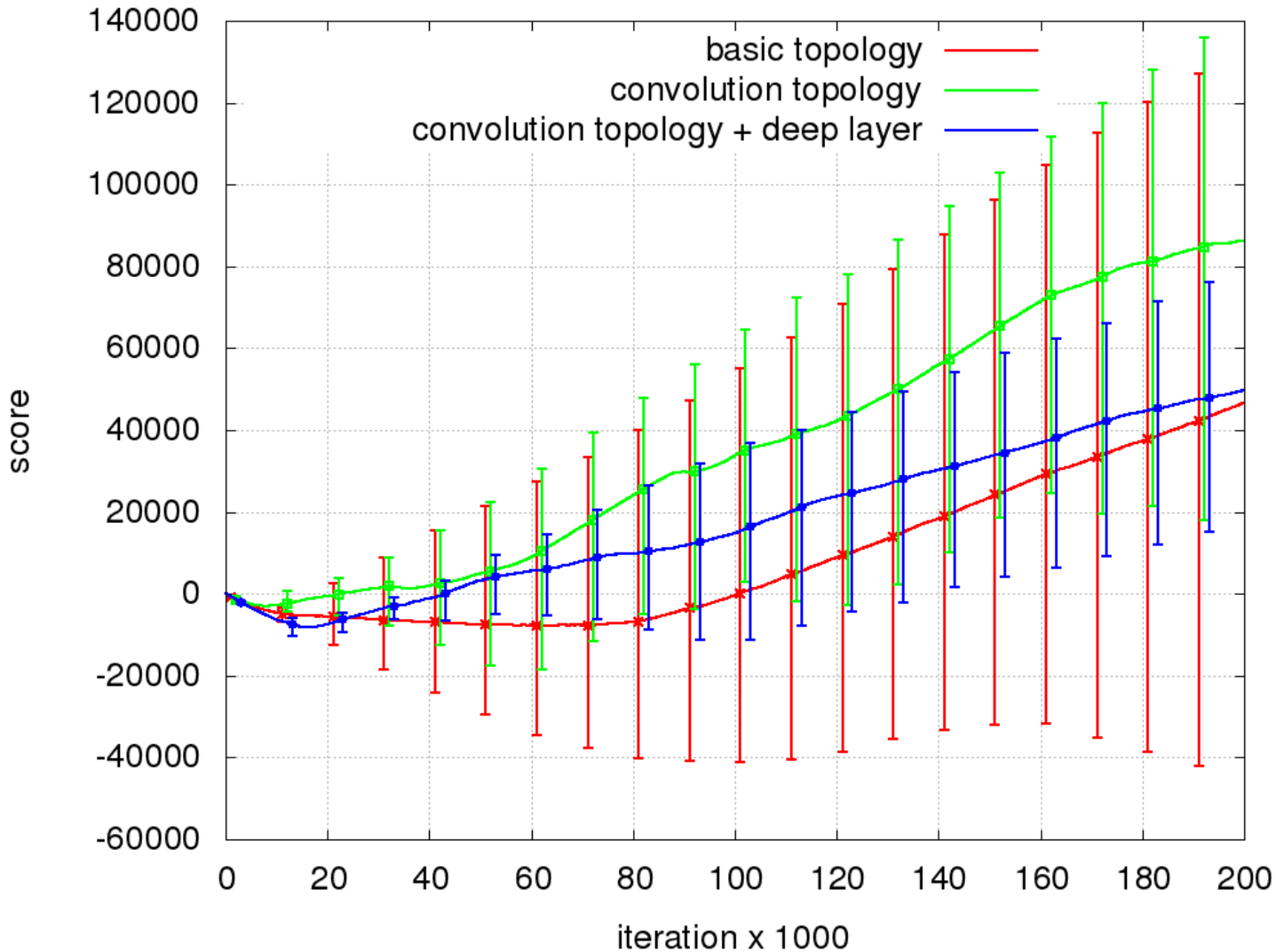
$$a(n) = \{-1, 0, 1\}$$
$$v(n) = \alpha r(n-1) + (1.0 - \alpha)a(n)$$
$$p(n) = p(n-1) + v(n)$$



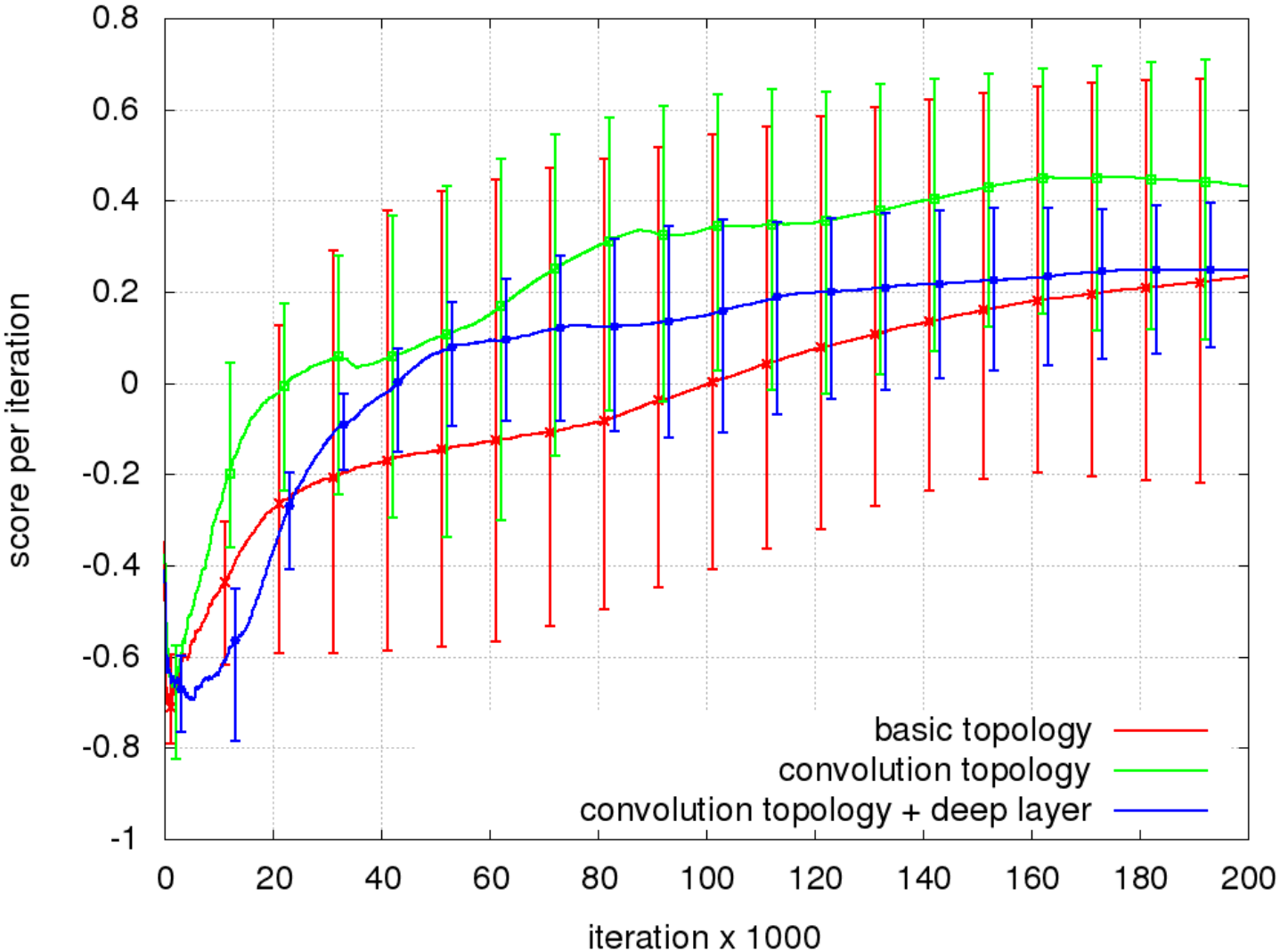

- reward

$$r(n) = \begin{cases} +1, & \text{if green field} \\ -1, & \text{otherwise} \end{cases}$$

# Line follower - results



# Line follower - results



# Q&A

https://github.com/michalnand/robotics

https://github.com/michalnand/machine_learning_new

michal.nand@gmail.com